Minh Dân Nguyễn @bill98 Follow
37 1 2

Published May 4th, 5:34 PM - 5 min read
736 4 2

Giải đáp thắc mắc về Variable và bộ nhớ trong Python

MayFest ViblоMayFest memory variable Interview

Có thể bạn chưa biết: Trong tháng 5 này 300 thành viên đầu tiên hoàn thành 4 bài viết hợp lệ sẽ nhận được bộ phần quà bao gồm: 1 Áo phông, 1 Túi, Stickers. 👉 **Đăng ký ngay tại đây.**

Mở Đầu

Đầu tiên phải nói rằng mình đã có rất nhiều câu hỏi về bộ nhớ, biến được lưu trữ trong Python. Đôi khi nó làm mình cảm thấy không hiểu gì và không rành mạch hơn các ngôn ngữ trước đây mình từng tìm hiểu như C hay Java. Do đó khi gặp các câu hỏi này lần đầu, mình đã chạy code trên máy cho chắc ăn, xem nó như thế nào (Chứ không dám chạy code trong đầu 😊) , dùng các trang để visualize để có thể hiểu thêm về cách chạy , lưu chữ của chúng. Bắt đầu thôi.

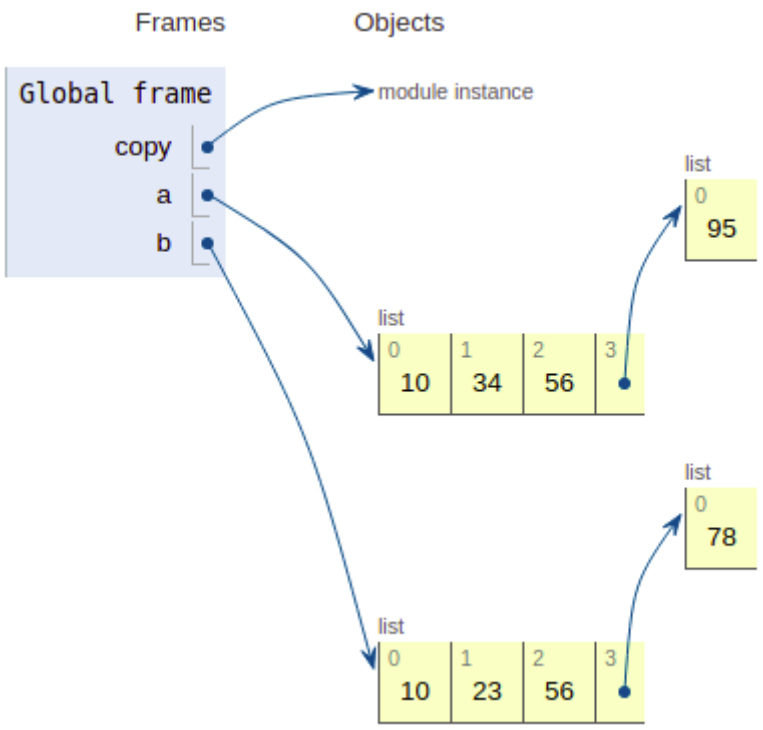
Câu số 1: Copy một list trong Python

```
import copy
a = [10, 23, 56, [78]]
b = copy.deepcopy(a)
a[3][0] = 95
a[1] =34
# List b = ???
```

Câu hỏi này gần như có ở tất cả các bài Test về Python .

List b sẽ ra sao trong tình huống này ?

CÂU TRẢ LỜI LÀ : b SẼ KHÔNG THAY ĐỔI ! Giải thích : deepcopy(sao chép sâu) có nghĩa là nó tạo ra một bản sao đối tượng, chèn các đối tượng được tìm thấy trong bản gốc vào nó ! Do đó có bất kỳ thay đổi ở đối tượng ban đầu như thế nào cũng sẽ không ảnh hưởng đến đối tượng được hình thành sau khi copy (ở đây là b).

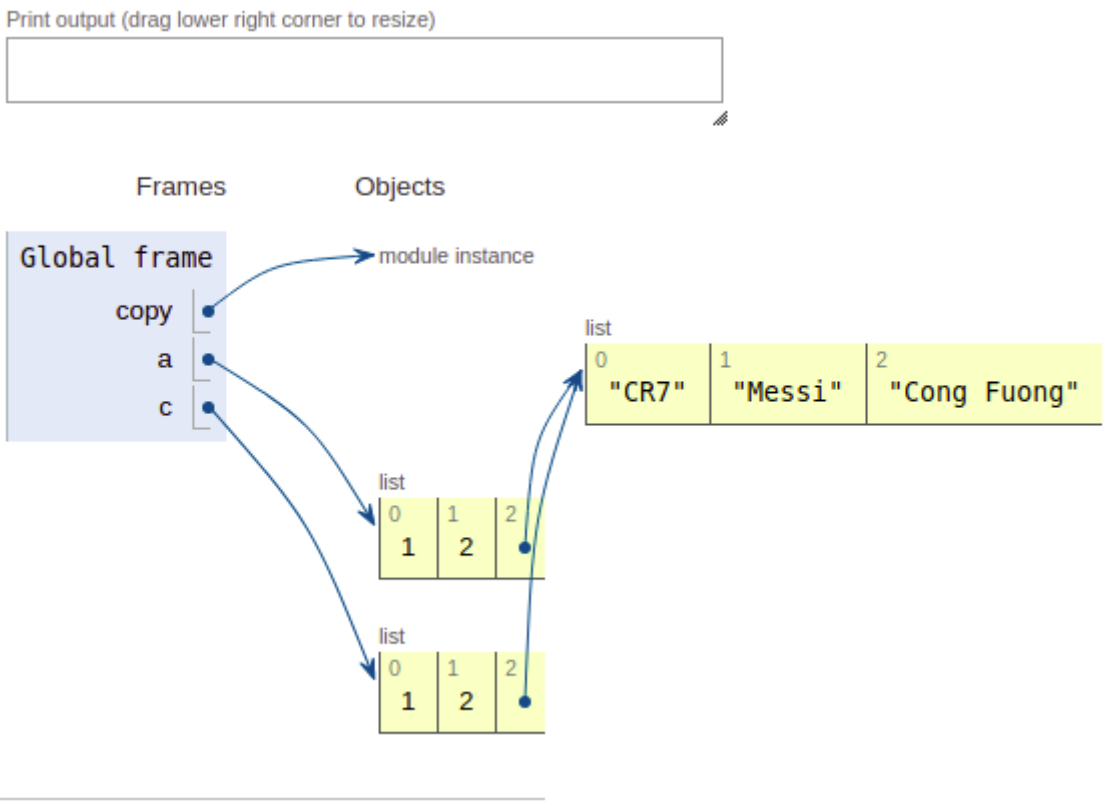


Hình trên mô tả rất rõ ràng về DEEP COPY.

Nhân tiện câu hỏi này, mình xin bổ sung thêm về SHALLOW COPY

```
import copy
a = [1, 2, ['CR7', 'Messi']]
c = copy.copy(a)
c[2].append('Cong Fuong')
print(a)
```

Giải thích về cách làm việc của Shallow copy : xây dựng một đối tượng mới sau đó chèn vào nó các tham chiếu cho các đối tượng được tìm thấy trong bản gốc.

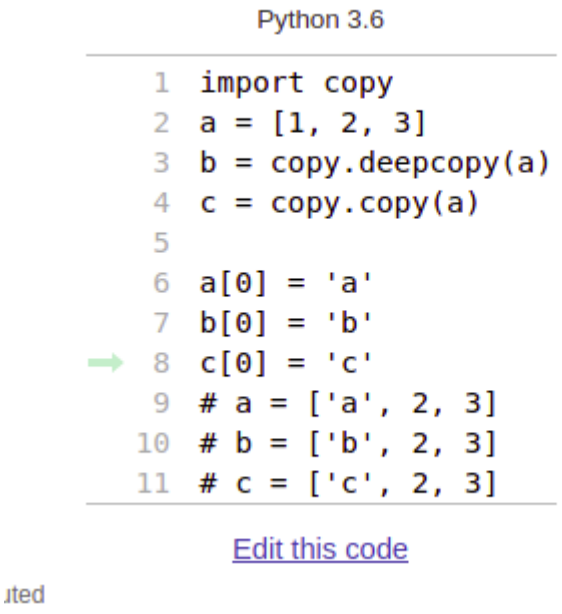


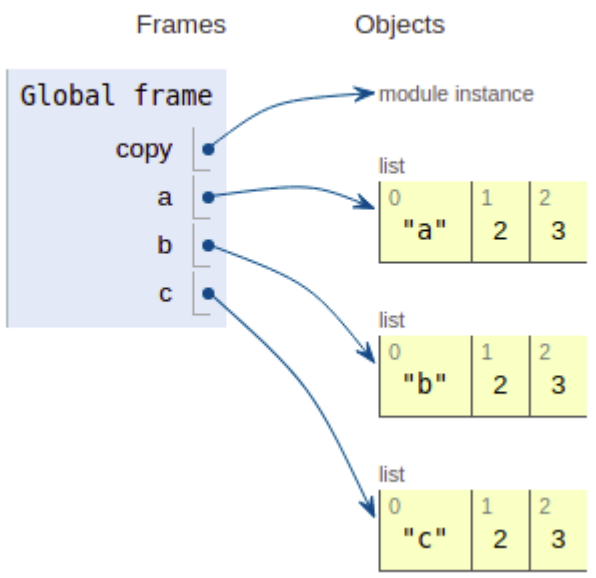
Hình ảnh trên đã rõ câu trả lời) 😃)

Một lưu ý hết sức quan trọng : ***SỰ KHÁC BIỆT GIỮA SAO CHÉP NÔNG VÀ SÂU CHỈ KHI CÓ SỰ LIÊN QUAN ĐẾN CÁC ĐỐI TƯỢNG GHÉP (ĐỐI TƯỢNG CHỨA ĐỐI TƯỢNG) NHƯ DANH SÁCH ... HAY CLASS INSTANCES-***

Cụ thể là phần list con trong list cha như ví dụ trên `a[2] : ['CR7', 'Messi']`

Còn nếu như ví dụ ở dưới đây thì sự khác biệt là không có nhé :





Câu số 2: Liên quan đến gán trong Python

Câu hỏi :

```
a = [1, 2, 3]
b = a
a.append('日本語')
b.append('English')
# b = ?
```

Đáp án sẽ là : Cả a và b đều có kết quả là : [1, 2, 3, '日本語', 'English'] .

Giải thích : Tưởng rằng b sẽ tạo ra một danh sách riêng biệt: a, b có thể thay đổi tùy ý và độc lập. Nhưng không , thay đổi ở cả 2 đều tác động lên nhau.

Về mặt bộ nhớ : Khi gán một biến (a = [1,2,3]), chúng ta chỉ nó đến một địa chỉ ô nhớ. Khi đó a là một dạng biến trỏ đến địa chỉ bộ nhớ của đối tượng mới (là [1,2,3]) Và như hình dưới đây cả hai đều cùng trỏ vào một ô nhớ !!

```
print(a, hex(id(a)))
print(b, hex(id(b)))
#[1, 2, 3, '日本語', 'English'] 0x7fe6e5ab1f48
#[1, 2, 3, '日本語', 'English'] 0x7fe6e5ab1f48
```

Câu số 3: Liên quan thay đổi chuỗi

```
a = 'jose'
b = a
print(hex(id(a)), hex(id(b)))
# a: 0x7f6d21920fb8  b :0x7f6d21920fb8
b += 'mourinho'
# a: 0x7f6d21920fb8 b: 0x7f6d1f91eb30
print(a) # jose
print(b) # josemourinho
```

Chúng ta sẽ đặt câu hỏi : Tại sao a, b cũng như câu hỏi trên cùng tham chiếu đến một ô nhớ : Tại sao khi b thay đổi mà a lại không thay đổi theo giống như câu hỏi ở trên kia. Điều đó được giải thích qua Mutable vs Immutable Objects trong Python. String là dạng Immutable , do đó nó không cho phép thay đổi giá trị của bản thân nó. Giá trị tại ô nhớ ban đầu là b :0x7f6d21920fb8 vẫn mãi mãi có giá trị là : 'jose'.

Sau lệnh b += 'mourinho' thì địa chỉ ô nhớ b trỏ đến đã thay đổi : :0x7f6d21920fb8 -> 0x7f6d1f91eb30

Còn về ví dụ List ở trên , nó cho phép chúng ta thay đổi giá trị của bản thân nó (tại chính ô nhớ đó) . Vì list là dạng Mutable objects. Xem hình dưới đây :

```
a = ['jose']
b = a
print(id(a), id(b))
# a: 0x7f280c89ef48  b: 0x7f280c89ef48
b[0] = 'jose mourinho'
# a: 0x7f280c89ef48  b: 0x7f280c89ef48
print(a) # ['jose mourinho']
print(b) # ['jose mourinho']
```

Rõ ràng tại ô nhớ của b, giá trị đã được thay đổi , còn địa chỉ ô nhớ thì không hề thay đổi nhé .

Câu số 4: Liên quan đến so sánh == và IS

```
a = [1,2,4]
b = [1,2,4]
print(a is b) # False
print( a == b) # True
```

Giải thích : == là toán tử so sánh giá trị của 2 toán hạng, is là toán tử kiểm tra xem cả 2 toán hạng có cùng tham chiếu đến cùng một đối tượng hay không.

Ở đây a, b có giá trị bằng nhau nhưng không tham chiếu đến cùng một đối tượng.

Kết tạm :

Mutable vs Immutable Objects là một phần rất quan trọng đọc ở đây : https://medium.com/@meghamohan/mutable-and-immutable-side-of-python-c2145cf72747

Tài liệu về Python (chuẩn nhất) xem ở đây : https://docs.python.org/3/library/copy.html

Trang web để visual code Python : http://www.pythontutor.com/visualize.html#mode=edit

Related

Kiến thức phỏng vấn iOS _ Phần 1 : Structures and Classes
Lê Văn Tuấn
11 min read
3470 7 3 8

Memory Management in Swift - Quản lý bộ nhớ trong Swift (Phần 1)
vietanh
4 min read
1072 6 0 1

Sự khác biệt giữa kiểu dữ liệu nguyên thủy và kiểu dữ liệu đối tượng
Nguyen Van Bac
6 min read
4828 3 1 7

How and when override equals() and hashCode()
Ngo Duy
3 min read
2646 2 0 3

[Bắt đầu làm quen về hệ gợi ý (Recommender System)]

Minh Dân Nguyễn
4 min read
71 0 0 2

Comments

Login to comment

Alex Nguyen @alex-2201 May 4th, 11:31 PM

hay phết

0 | Reply Share •••

Minh Dân Nguyễn @bill98 May 4th, 11:37 PM

cảm ơn bạn nhé !! m ko ngờ là có người đọc 😃))

0 | Reply Share •••

Sophia Phan @phanthanhthuy May 6th, 4:32 PM

It's so helpful. Thank for your sharing bro!

0 | Reply Share •••

Minh Dân Nguyễn @bill98 May 6th, 5:30 PM

Nice 😃))

0 | Reply Share •••